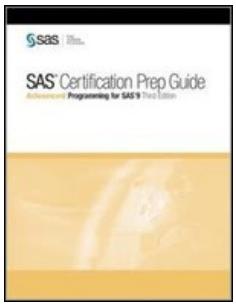

SAS Certification Prep Guide: Advanced Programming for SAS 9, Third Edition

by SAS Institute

Chapter 7: Creating and Managing Views Using PROC SQL

Overview

Introduction

A PROC SQL view is a stored query expression that reads data values from its underlying files, which can include SAS data files, DATA step views, other PROC SQL views, or DBMS data.

You can refer to views in queries as if they were tables. The view derives its data from the tables or views that are listed in its FROM clause. The data that is accessed by a view is a subset or superset of the data that is in its underlying table(s) or view(s).

```
proc sql;
   create view sasuser.raisev as
      select empid, jobcode,
             salary format=dollar12.2,
             salary/12 as MonthlySalary
             format=dollar12.2
         from payrollmaster;
   select *
      from sasuser.raisev
      where jobcode in ('PT2','PT3');
```

EmpID	JobCode	Salary	MonthlySalary
1333	PT2	$124,048.00	$10.337.33
1404	PT2	$127.926.00	$10,660.50
1118	PT3	$155.931.00	$12.994.25
1410	PT2	$118.559.00	$9,879.92
1777	PT3	$153.482.00	$12.790.17
1106	PT2	$125.485.00	$10, 457.08
1442	PT2	$118.350.00	$9,862.50
1478	PT2	$117,884.00	$9,823.67
1890	PT2	$120.254.00	$10,021.17
1107	PT2	$125.968.00	$10,497.33
1830	PT2	$118.259.00	$9.854.92
1928	PT2	$125.801.00	$10,483.42

PROC SQL views

- can be used in SAS programs in place of an actual SAS data file
- can be joined with tables or other views
- can be derived from one or more tables, PROC SQL views, or DATA step views
- can access data from a SAS data set, a DAT A step view, a PROC SQL view, or a relational database table
- extract underlying data, which enables you to access the most current data.

Objectives

In this chapter, you leam to

- create and use PROC SQL views

- display the definition for a PROC SQL view
- manage PROC SQL views
- update PROC SQL views
- drop (delete) PROC SQL views.

Prerequisites

Before beginning this chapter, you should complete the following chapters:

- "Performing Queries Using PROC SQL" on page 4
- "Performing Advanced Queries Using PROC SQL" on page 29
- "Combining Tables Horizontally Using PROC SQL" on page 86
- "Creating and Managing Tables Using PROC SQL" on page 175.

Creating and Using PROC SQL Views

PROC SQL Views

A PROC SQL view is a stored query that is executed when you use the view in a SAS procedure or DATA step. A view contains only the descriptor and other information required to retrieve the data values from other SAS files (SAS data files, DATA step views, or other PROC SQL views) or external files (DBMS data files). The view contains only the logic for accessing the data, *not* the data itself.

Because PROC SQL views are not separate copies of data, they are referred to as *virtual tables*. They do not exist as independent entities like real tables. However, views use the same naming conventions as tables and can be used in SAS programs in place of an actual SAS table. Like tables, views are considered to be SAS data sets.

Views are useful because they

- often save space (a view is usually quite small compared with the data that it accesses)
- prevent users from continually submitting queries to omit unwanted columns or rows
- ensure that input data sets are always current, because data is derived from tables at execution time
- shield sensitive or confidential columns from users while enabling the same users to view other columns in the same table
- hide complex joins or queries from users.

Creating PROC SQL Views

You use the CREATE VIEW statement to create a view.

General form, CREATE VIEW statement:

```
CREATE VIEW proc-sql-viewAS
    SELECT column-l<, … column-n>
         FROM table-1 \ view-l<, … table-n \ view-n>
         <optional query clauses>;
```

where

- *proc-sql-view* specifies the name of the PROC SQL view that you are creating.
- SELECT specifies the column(s) that will appear in the table.

- FROM specifies the table(s) or view(s) to be queried.
- *optional query clauses* are used to refine the query further and include the WHERE, GROUP BY, HAVING, and ORDER BY clauses.

A PROC SQL view derives its data from the tables or views that are listed in the FROM clause. The data that is accessed by a view is a subset or superset of the data that is in its underlying table(s) or view(s). When a view is referenced by a SAS procedure or in a DATA step, it is executed and, conceptually, an internal table is built. PROC SQL processes this internal table as if it were any other table.

Example

The following PROC SQL step creates a view that contains information for flight attendants. The view always returns the employee's age as of the current date.

The view *Sasuser.Faview* creates a virtual table from the accompanying SELECT statement. Although the underlying tables, *Sasuser.payrollmaster* and *Sasuser.Staffmaster*, can change, the instructions that comprise the view stay constant. The libref specified in the FROM clause is optional. It is assumed that the contributing tables are stored in the same library as the view itself, unless otherwise specified.

```
proc sql;
   create view sasuser.faview as
      select lastname, firstname, gender,
             int((today()-dateofbirth)/365.25) as Age,
             substr(jobcode,3,1) as Level,
             salary
         from sasuser.payrollmaster,
              sasuser.staffmaster
         where jobcode contains 'FA' and
               staffmaster.empid=
               payrollmaster.empid;
```

When this PROC SQL step is submitted, SAS does not actually execute the SELECT statement that follows the AS keyword, but partially compiles and stores the SELECT statement in a data file with a member type of VIEW. A message in the SAS log confirms that the view has been defined.

Table 7.1: SAS Log

```
1    proc sql;
2            create view sasuser.faview as
3              select lastname, firstname, gender,
4                     int((today()-dateofbirth)/365.25)
5                        as Age,
6                     substr(jobcode,3,1) as Level,
7                     salary
8                 from sasuser.payrollmaster,
9                      sasuser.staffmaster
10                where jobcode contains 'FA' and
11                     staffmaster.empid=
12                     payrollmaster.empid;
NOTE: SQL view SASUSER.FAVIEW has been defined.
```

Tip It is helpful to give a PROC SQL view a name that easily identifies it as a view. For example, *Faview* or *Fav*.

Note In the Windows and UNIX operating environments, the default extension for PROC SQL views (and DATA step views) is *.sas7bvew*.

Using PROC SQL Views

You can use a view in a subsequent PROC SQL step, or later in the same step, just as you would use an actual SAS table. In the following example, the PROC SQL view *Sasuser.Faview* is used in a query. Because the query stored in the view calculates the age of each flight attendant based on the current date, the resulting output from this PROC SQL step shows each flight attendant's age as of the current date. If *Sasuser.Faview* were a static table, instead of a view, the age shown

for each flight attendant would never change.

```
proc sql;
   select *
      from sasuser.faview;
```

Partial output is shown below.

LastName	FirstName	Gender	Age	Level	Salary
ARTHUR	BARBARA	F	47	3	$46,040
CAHILL	MARSHALL	M	52	2	$40,001
CARTER	DOROTHY	F	52	3	$46,346
COOPER	ANTHONY	M	53	3	$45,104
DEAN	SHARON	F	51	3	$46,787
DUNLAP	DONNA	F	44	2	$40,443
EATON	ALICIA	F	51	2	$38.902
FIELDS	DIANA	F	54	1	$32,448
FLETCHER	MARIE	F	48	1	$31,436
GOMEZ	ALAN	M	42	1	$31,175

Tip You can use PROC SQL views in other SAS procedures and DATA steps. In the following example, PROC TABULATE calculates the flight attendants' mean age by level, using the view *Sasuser.Faview*.

```
proc tabulate data=sasuser.faview;
   class level;
   var age;
   table level*age*mean;
run;
```

level		
1	2	3
Age	Age	Age
Mean	Mean	Mean
44.27	48.31	49.86

Note The values for the variable **Age** will vary, because the calculation is dependent on the date on which the code is executed.

Note For information about the TABULATE procedure, see the SAS documentation.

Displaying the Definition for a PROC SQL View

Overview

You can use a DESCRIBE VIEW statement to display a definition of a view in the SAS log.

General form, DESCRIBE VIEW statement:

DESCRIBE VIEW *proc-sql-view-1<,…proc-sql-view-n>;*

where

proc-sql-view

specifies a PROC SQL view and can be one of the following:

- a one-level name
- a two-level libref.view name
- a physical pathname that is enclosed in single quotation marks.

Tip If you use a PROC SQL view in a DESCRIBE VIEW statement that is based on or derived from another view, then you might want to use the FEEDBACK option in the PROC SQL statement. This option displays in the SAS log how the underlying view is defined and expands any expressions that are used in this view definition.

Example

The following PROC SQL step writes the view definition for *Sasuser.Faview* to the SAS log:

```
proc sql;
   describe view sasuser.faview;
```

Table 7.2: SAS Log

```
NOTE: SQL view SASUSER.FAVIEW is defined as:
      select lastname, firstname, gender,
             INT((TODAY()-dateofbirth)/365.25) as Age,
      SUBSTR(jobcode, 3, 1) as Level, salary
      from SASUSER.PAYROLLMASTER, SASUSER.STAFFMASTER
      where jobcode contains 'FA' and
            (staffmaster.empid=payrollmaster.empid);
```

Managing PROC SQL Views

Guidelines for Using PROC SQL Views

When you are working with PROC SQL views, it is best to follow these guidelines:

- Avoid using an ORDER BY clause in a view definition, which causes the data to be sorted every time the view is executed. Users of the view might differ in how or whether they want the data to be sorted, so it is more efficient to specify an ORDER BY clause in a query that references the view.
- If the same data is used many times in one program or in multiple programs, it is more efficient to create a table rather than a view because the data must be accessed at each view reference. (This table can be a temporary table in the *Work* library.)
- Avoid creating views that are based on tables whose structure might change. A view is no longer valid when it references a nonexistent column.
- If a view resides in the same SAS library as the contributing table(s), it is best to specify a one-level name in the FROM clause.

Omitting the Libref

The default libref for the table or tables in the FROM clause is the libref of the library that contains the view. Using a one-level name in the FROM clause prevents you from having to change the view if you assign a different libref to the SAS library that contains the view and its contributing table or tables.

The following PROC SQL step creates the view *Sasuser.Payrollv*. The FROM clause specifies a two-level name for the contributing table, *Sasuser.payrollmaster*. However, it is not necessary to specify the libref *Sasuser* because the contributing table is assumed to be stored in the same library as the view.

```
proc sql;
   create view sasuser.payrollv as
      select *
         from sasuser.payrollmaster;
```

When the one-level name *Payrollmaster* is used in the FROM clause, *Sasuser.payrollmaster* is being specified, though it

appears that *Work.Payrollmaster* is being specified.

```
proc sql;
   create view sasuser.payrollv as
      select *
         from payrollmaster;
```

Caution If you are creating a view that is stored in a different library than the table(s) referenced in the FROM clause, you must specify a two-level name for the table(s).

Using an Embedded LIBNAME Statement

As an alternative to omitting the libref in the FROM clause, you can embed a LIBNAME statement in a USING clause to store a SAS libref in a view. Embedding a LIBNAME statement is a more flexible approach because

- it can be used regardless of whether the view and the underlying tables reside in the same library
- it avoids the confusion that might arise if a libref is omitted from a table name in the FROM clause.

An embedded LIBNAME statement can be used only with a PROC SQL view. A libref created with an embedded LIBNAME statement will not conflict with an identically named libref in the SAS session.

General form, USING clause:

USING *libname-clause-1<,… libname-clause-n>;*

where

libname-clause

is one of the following:

- a valid LIBNAME statement
- a valid SAS/ACCESS LIBNAME statement.

Caution The USING clause must be the last clause in the CREATE VIEW statement.

Example

In the following example, while the view *Sasuser.Payrollv* is executing in the PROC PRINT step, the libref *Airline* is dynamically assigned in the USING clause.

```
proc sql;
   create view sasuser.payrollv as
      select*
         from airline.payrollmaster
         using libname airline 'SAS-library-one';
quit;
proc print data=sasuser.payrollv;
run;
```

If an earlier assignment of the libref AIRLINE exists, the EMBEDDED LIBNAME statement overrides the assignment for the duration of the view's execution. After the view executes, the original libref assignment is reestablished and the embedded assignment is cleared.

Creating a View to Enhance Table Security

One advantage of PROC SQL views is that they can bring data together from separate sources. This enables views to be used to shield sensitive or confidential columns from some users while enabling the same users to view other columns in the same table.

Caution Although PROC SQL views can be used to enhance table security, it is strongly recommended that you *use the security features that are available in your operating environment to maintain table security*.

Example

The following PROC SQL step creates the view *Manager.Infoview*. The view accesses data about flight attendants that is stored in three SAS libraries: *Fal, Fa2*, and *Fa3*. The *Fal, Fa2, and Fa3* libraries can be assigned access privileges at the operating system level to prevent

- Level 1 flight attendants from reading the data stored in the *Fa2* and *Fa3* libraries
- Level 2 flight attendants from reading the data stored in the *Fal* and *Fa3* libraries
- Level 3 flight attendants from reading the data stored in the *Fal* and *Fa2* libraries.

Access privileges can also be assigned to permit managers (who are authorized to access all SAS libraries) to view all of the information.

```
proc sql;
   create view manager.infoview as
      select *
         from fa1.info
      outer union corr
      select *
         from fa2.info
      outer union corr
      select *
         from fa3.info;
```

Updating PROC SQL Views

Overview

You can update the data underlying a PROC SQL view using the INSERT, DELETE, and UPDATE statements under the following conditions:

- You can update only a single table through a view. The table cannot be joined or linked to another table, nor can it contain a subquery.
- You can update a column using the column's alias, but you cannot update a derived column (a column that is produced by an expression).
- You can update a view that contains a WHERE clause. The WHERE clause can be specified in the UPDATE clause or in the view. You cannot update a view that contains any other clause such as an ORDER BY or a HAVING clause.
- You cannot update a summary view (a view that contains a GROUP BY clause).

Updating a view does *not* change the stored instructions for the view. Only the data in the underlying table(s) is updated.

Example

The following PROC SQL step creates the view *Sasuser.Raisev*, which includes the columns **`Salary`** and **`MonthlySalary.`** A subsequent query that references the view shows the columns.

```
proc sql;
   create view sasuser.raisev as
      select empid, jobcode,
             salary format=dollar12.,
             salary/12 as MonthlySalary
             format=dollar12.
         from payrollmaster;

proc sql;
   select *
      from sasuser.raisev
      where jobcode in ('PT2','PT3');
```

EmpID	JobCode	Salary	MonthlySalary
1333	PT2	$124.048	$10.337
1404	PT2	$127.926	$10.661
1118	PT3	$155.931	$12,994
1410	PT2	$118.559	$9,880
1777	PT3	$153.482	$12.790
1106	PT2	$125.485	$10,457
1442	PT2	$118,350	$9.863
1478	PT2	$117.884	$9.824
1890	PT2	$120.254	$10.021
1107	PT2	$125.968	$10.497
1830	PT2	$118.259	$9.855
1928	PT2	$125.801	$10.483

Suppose you want to update the view to show a salary increase for employees whose job code is *PT3*. You can use an UPDATE statement to change the column **Salary** and a WHERE clause in the UPDATE clause to identify the rows where the value of **JobCode** equals *PT3*. Though **MonthlySalary** is a derived column and cannot be changed using an UPDATE statement, it will be updated because it is derived from **Salary.**

When the PROC SQL step is submitted, a note appears in the SAS log that indicates how many rows were updated:

```
proc sql;
   update sasuser.raisev
      set salary=salary * 1.20
      where jobcode='PT3';
```

Table 7.3: SAS Log

```
116   proc sql;
117      update sasuser.raisev
118         set salary=salary * 1.20
119         where jobcode='PT3';
NOTE: 2 rows were updated in SASUSER.RAISEV.
```

Note Remember that the rows were updated in the *table* that underlies the view *Sasuser.Raisev*.

When you resubmit the query, the updated values for **Salary** and **MonthlySalary** appear in the rows where **JobCode** equals *PT3*

```
proc sql;
   select *
      from sasuser.raisev
      where jobcode in ('PT2','PT3');
```

EmpID	JobCode	Salary	MonthlySalary
1333	PT2	$124,048	$10,337
1404	PT2	$127,926	$10,661
1118	PT3	$187,117	$15.593
1410	PT2	$118,559	$9,880
1777	PT3	$184.178	$15,348
1106	PT2	$125,485	$10,457
1442	PT2	$118,350	$9,863
1478	PT2	$117,884	$9,824

1890	PT2	$120,254	$10,021
1107	PT2	$125,968	$10 497
1830	PT2	$118,259	$9,855
1928	PT2	$125,801	$10,483

Dropping PROC SQL Views

Overview

To drop (delete) a view, use the DROP VIEW statement.

General form, DROP VIEW statement:
DROP VIEW *view-name-1 <,…view-name-n>;*

where

view-name

specifies a SAS data view of any type (PROC SQL view or DATA step view) and can be one of the following:

- a one-level name
- a two-level *Hbref.view* name
- a physical pathname that is enclosed in single quotation marks.

Example

The following PROC SQL step drops the view *Sasuser.Raisev*. After the step is submitted, a message appears in the SAS log to confirm that the view has been dropped.

```
proc sql;
   drop view sasuser.raisev;
```

Table 7.4: SAS Log

```
21   proc sql;
22      drop view sasuser.raisev;
NOTE: View SASUSER.RAISEV has been dropped.
```

Summary

Contents

This section contains the following topics.

Text Summary

Using PROC SQL Views

A PROC SQL view is a stored query that is executed when you use the view in a SAS procedure or DATA step. A view

contains only the descriptor and other information required to retrieve the data values from other SAS files (SAS data files, DATA step views, or other PROC SQL views) or external files (DBMS data files). When executed, a PROC SQL view's output can be a subset or superset of one or more underlying files. A view contains no data, but describes or defines data that is stored elsewhere.

PROC SQL views

- can be used in SAS programs in place of an actual SAS data file
- can be joined with tables or other views
- can be derived from one or more tables, PROC SQL views, DATA step views, or SAS/ACCESS views.
- extract underlying data, which enables you to access the most current data.

Because PROC SQL views are not separate copies of data, they are referred to as virtual tables. They do not exist as independent entities like real tables. However, views use the same naming conventions as tables and can be used in SAS programs in place of an actual SAS table. Like tables, views are considered to be SAS data sets.

Creating SQL Views

You use the CREATE VIEW statement to create a view. A PROC SQL view derives its data from the tables or views that are listed in the FROM clause. The data that is accessed by a view is a subset or superset of the data that is in its underlying tables(s) or view(s). When a view is referenced by a SAS procedure or in a DATA step, it is executed and, conceptually, an internal table is built. PROC SQL processes this internal table as if it were any other table. A view can be used in a subsequent PROC SQL step just as you would use an actual SAS table.

Displaying the Definition for a PROC SQL View

You can use a DESCRIBE VIEW statement to display a definition of a view in the SAS log.

Managing PROC SQL Views

The default libref for the table or tables in the FROM clause is the libref of the library that contains the view. Using a one-level name prevents you from having to change the view if you assign a different libref to the SAS library that contains the view and its contributing table or tables.

As a more flexible alternative to omitting the libref in the FROM clause, you can embed a LIBNAME statement in a USING clause if you want to store a SAS libref in a view. Embedding a LIBNAME statement in a USING clause does not conflict with an identically named libref in the SAS session.

One advantage of PROC SQL views is that they can bring data together from separate sources. This enables views to be used to shield sensitive or confidential columns from some users while enabling the same users to view other columns in the same table. Although PROC SQL views can be used to enhance table security, it is strongly recommended that you use the security features that are available in your operating environment to maintain table security.

Updating PROC SQL Views

You can update the data underlying a PROC SQL view using the INSERT, DELETE, and UPDATE statements under the following conditions:

- You can update only a single table through a view. The table cannot be joined or linked to another table, nor can it contain a subquery.
- You can update a column using the column's alias, but you cannot update a derived column (a column that is produced by an expression).
- You can update a view that contains a WHERE clause. The WHERE clause can be in the UPDATE clause or in the view. You cannot update a view that contains any other clause such as an ORDER BY or a HAVING clause.
- You cannot update a summary view (a view that contains a GROUP BY clause).

Dropping PROC SQL Views

To drop (delete) a view, use the DROP VIEW statement.

Syntax

```
PROC SQL;
     CREATE VIEW proc-sql-view AS
         SELECT column-l<, … column-n>
             FROM table-1 | view-l<, … table-n | view-n>
             <optional query clauses>;
             USING libname-clause<,…libname-clause>;
     DESCRIBE VIEW proc-sql-view<,…proc-sql-view>;

INSERT INTO table-name | proc-sql-view
             <(target-column-l<, … target-column-n)>
          SET column-l=value-l<, … column-n=value-n>
          <… SETcolumn-l=value-l<, … column-n=value-n>>;
     DELETE FROM table-name | proc-sql-view
          <WHEREexpression>;
     UPDATE table-name \ proc-sql-view
          SET column-l=expression<, … column-n=expression>>
          <WHEREexpression>;
     DROP VIEW view-name<,…view-name>;

   QUIT;
```

Sample Programs

Creating a PROC SQL View

```
proc sql;
   create view sasuser.raisev as
      select empid, jobcode,
             salary format=dollar12.2,
             salary/12 as MonthlySalary
             format=dollar12.
         from payrollmaster
         using libname airline 'c:\data\ia';
quit;
```

Displaying the Definition for a PROC SQL View

```
proc sql;
   describe view sasuser.raisev;
quit;
```

Using a PROC SQL View in a Query

```
proc sql;
   select *
      from sasuser.raisev
      where jobcode in ('PT2','PT3');
quit;
```

Updating a PROC SQL View

```
proc sql;
   update sasuser.raisev
      set salary=salary * 1.20
      where jobcode='PT3';
quit;
```

Dropping a PROC SQL View

```
proc sql;
   drop view sasuser.raisev;
quit;
```

Points to Remember

- Avoid using an ORDER BY clause in a view definition, which causes the data to be sorted every time the view is executed. Users of the view might differ in how or whether they want the data to be sorted, so it is more efficient to specify an ORDER BY clause in a query that references the view.

- If the same data is used many times in one program or in multiple programs, it is more efficient to create a table rather than a view because the data must be accessed at each view reference. (This table can be a temporary table in the *Work* library.)
- Avoid creating views that are based on tables whose structure might change. A view is no longer valid when it references a nonexistent column.
- If a view resides in the same SAS library as the contributing table(s), it is best to specify a one-level name in the FROM clause.

Quiz

Select the best answer for each question. After completing the quiz, check your answers using the answer key in the appendix.

1. Which of the following statements is false regarding a PROC SQL view? ?

a. A view cannot be used in ajoin.

b. A view accesses the most current underlying data.

c. A view follows the same naming conventions as a table.

d. A view can be used in SAS programs in place of an actual SAS data file.

2. Which of the following statements describes an advantage of using a PROC SQL view? ?

a. Views often save space, because a view is usually quite small compared with the data that it accesses.

b. Views prevent users from continually submitting queries to omit unwanted columns or rows.

c. Views hide complex joins or queries from users.

d. all of the above

3. Which PROC SQL step creates a view that queries the table *Sasuser.payrollmaster?* ?

a.
```
a.  proc sql;
     insert into sasuser.newview
        select * from sasuser.payrollmaster;
```

b.
```
b.  proc sql;
     create sasuser.newview as
        select * from sasuser.payrollmaster;
```

c.
```
c.  proc sql;
     create view sasuser.newview as
        select * from sasuser.payrollmaster;
```

d.
```
d.  proc sql;
     select * from sasuser.payrollmaster
        into view sasuser.newview;
```

4. Which of the following PROC SQL steps enables you to see a description of the view definition? ?

a.
```
a.  proc sql;
     select * from sasuser.payrollmasterv;
```

b.
```
b.  proc sql;
     describe view sasuser.payrollmasterv;
```

c.
```
c.  proc sql;
     list sasuser.payrollmasterv;
```

d.
```
d.  proc sql;
     contents view=sasuser.payrollmasterv;
```

5. Which PROC SQL step correctly references the view *Data.Empview?* ?

a. a.
```
proc sql;
 select *
     from data.empview;
```

b. b.
```
proc sql;
  select *
    from view data.empview;
```

c. c.
```
proc sql;
 select view *
     from data.empview;
```

d. d.
```
proc sql;
  select *
    from data
    where view='empview';
```

6. Which of the following PROC SQL steps correctly embeds a LIBNAME statement with a view definition? ?

a. a.
```
proc sql;
 insert into sasuser.newview
    select * from airline.supervisors
       libname airline 'c:\mysql';
```

b. b.
```
proc sql;
 create view sasuser.newview as
     from airline.supervisors
        embed libname airline 'c:\mysql';
```

c. c.
```
proc sql;
 using airline 'c:\mysql';
    insert into sasuser.newview
       select * from airline.supervisors;
```

d. d.
```
proc sql;
  create view sasuser.newview as
     select * from airline.supervisors
        using libname airline 'c:\mysql';
```

7. PROC SQL views can access data from ?

a. a SAS data file.

b. another PROC SQL view.

c. a relational database table.

d. all of the above

8. When you are working with PROC SQL views, it is best to ?

a. avoid using an ORDER BY clause in a view.

b. avoid creating views that are based on tables whose structure might change.

c. specify a one-level name in the FROM clause if the view resides in the same SAS library as the contributing table(s).

d. all of the above

9. You can update the data underlying PROC SQL view using the INSERT, DELETE, and UPDATE statements ?

under which of the following conditions:

a. The view is joined or linked to another table.

b. The view contains a subquery.

c. The view contains a WHERE clause.

d. all of the above

10. Which of the following programs drops (deletes) a view? ?

a.
```
a.  proc sql;
     delete sasuser.newview;
```

b.
```
b.  proc sql;
     drop view sasuser.newview;
```

c.
```
c.  proc sql;
     erase view sasuser.newview;
```

d.
```
d.  proc sql;
     remove newview from sasuser;
```

Answers

1. Correct answer: a

A PROC SQL view accesses the most current underlying data and can be joined with tables or other views. In addition, a PROC SQL view can

- be used in SAS programs in place of an actual SAS data file
- be derived from one or more tables, PROC SQL views, or DATA step views.

2. Correct answer: d

PROC SQL views are useful because they

- often save space (a view is usually quite small compared with the data that it accesses)
- prevent users from continually submitting queries to omit unwanted columns or rows
- hide complex joins or queries from users.

In addition, PROC SQL views

- ensure that input data sets are always current, because data is derived from tables at execution time
- can be used to shield sensitive or confidential columns from users while enabling the same users to view other columns in the same table.

3. Correct answer: c

You use the CREATE VIEW statement to create a view. The keywords CREATE VIEW are followed by the name of the view and the keyword AS.

4. Correct answer: b

The DESCRIBE VIEW statement displays the view definition in the SAS log.

5. Correct answer: a

A view can be used in a PROC SQL step just as you would use an actual SAS table.

6. Correct answer: d

The USING clause enables you to embed a LIBNAME statement in your view definition. The USING clause must be the last clause in the CREATE VIEW statement.

7. Correct answer: d

PROC SQL views can access data from a SAS data file, a DATA step view, a PROC SQL view, or a relational database table.

8. Correct answer: d

When you are working with PROC SQL views, it is best to

- avoid using an ORDER BY clause in a view. If you specify an ORDER BY clause, the data must be sorted each time the view is referenced.
- avoid creating views that are based on tables whose structure might change. A view is no longer valid when it references a nonexistent column.
- specify a one-level name in the FROM clause if the view resides in the same SAS data library as the contributing table(s). Using a one-level name in the FROM clause prevents you from having to change the view if you assign a different libref to the SAS data library that contains the view and its contributing table or tables

9. Correct answer: c

You can update a PROC SQL view provided that the view does not join or link to another table, the view does not have a subquery, or you try to update a derived column. You can update a view that contains a WHERE clause. The WHERE clause can be in the UPDATE clause or in the view. You cannot update a view that contains any other clause such as an ORDER BY or a HAVING clause.

10. Correct answer: b

The DROP VIEW statement drops a view from the specified library.